UCF Center for Research in Computer Vision

CAP6412
Advanced Computer Vision

Mubarak Shah

shah@crcv.ucf.edu

HEC-245

Lecture-11: Diffusion Models-Introduction Part-II

Generative AI (Computer Vision)

- **Text**
  - Large Language Models (LLM)
    - Generate/Predict the next word
    - Auto-regressive
  - **Visual Language Models (VLM)/Large Multimodal Models (LMM)**

- **Images**
  - DALL-E
    - Auto-regressive model for image Generation
  - **Diffusion Models**
    - **Generate Images**, Videos, audio, …

Video Generation

https://x.com/minchoi

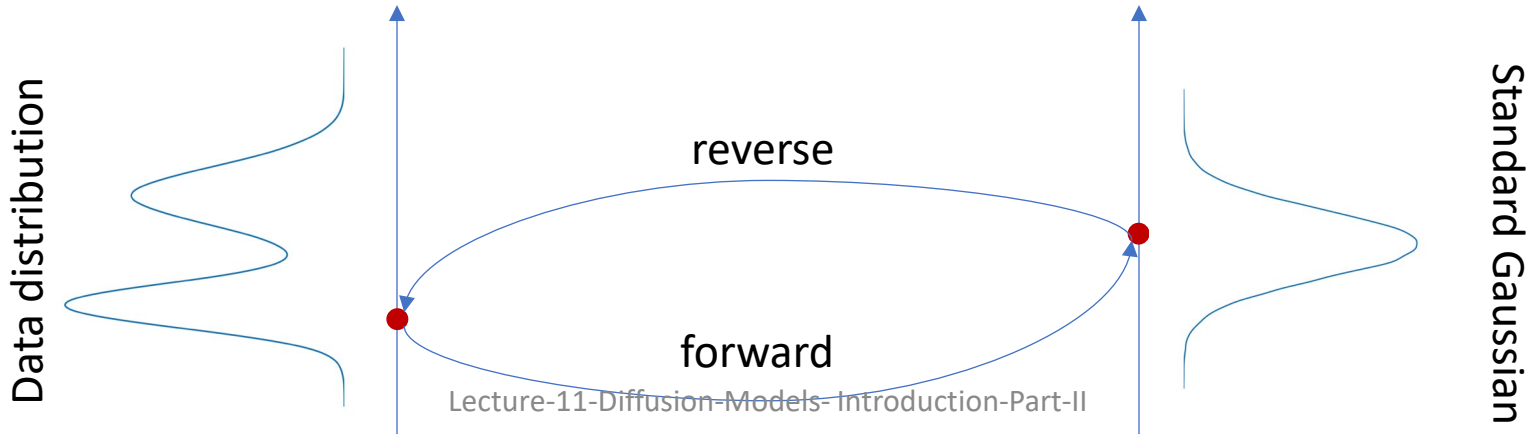
https://x.com/EHuanglu

Outline

1. Motivation
2. High-level overview
3. Denoising diffusion probabilistic models
4. Noise Conditioned Score Network

High-level overview

- Diffusion models are probabilistic models used for image generation
- They involve reversing the process of gradually degrading the data
- Consist of two processes:
  - **The forward process**: data is progressively destroyed by adding noise across multiple time steps
  - **The reverse process**: using a neural network, noise is sequentially removed to obtain the original data



Denoising **D**iffusion **P**robabilistic **M**odels (DDPMs)

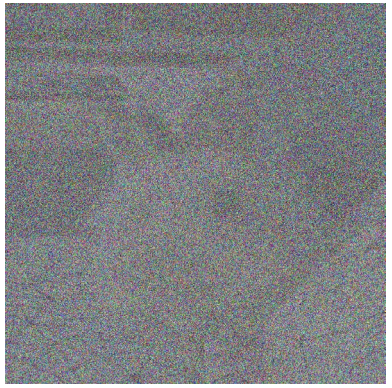
Forward process

x_0

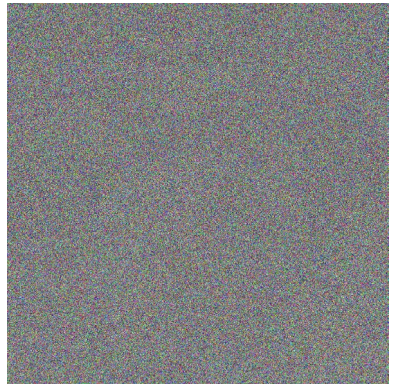


$x_0 \sim q(x)$



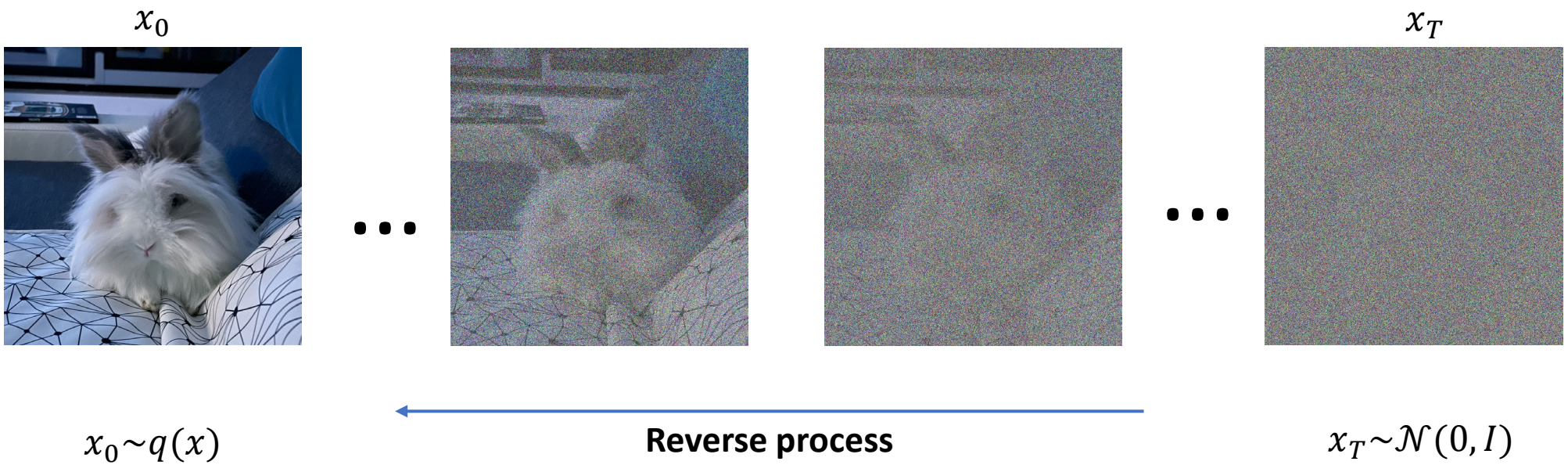


x_T



$x_T \sim \mathcal{N}(0, I)$

Denoising **D**iffusion **P**robabilistic **M**odels (DDPMs)



DDPMs. Training Algorithm

$$\min_{\theta} \underbrace{\frac{1}{T}\sum_{t=1}^{T} \mathbb{E}_{x_0 \sim q(x_0), z_t \sim \mathcal{N}(0,\mathrm{I})} \|z_t - z_\theta(x_t, t)\|_2^2}_{\mathcal{L}_{simple}}$$

Training algorithm:

Repeat

$x_0 \sim q(x_0)$

$t \sim \mathcal{U}(\{1, \dots, T\})$

$z_t \sim \mathcal{N}(0, \mathrm{I})$

$x_t = \sqrt{\hat{\beta}_t} \cdot x_0 + \sqrt{(1 - \hat{\beta}_t)} z_t$

$\theta = \theta - lr \cdot \nabla_\theta \mathcal{L}_{simple}$

Until convergence

$$\hat{\beta}_t = \prod_{i=1}^{t} \alpha_i$$

DDPMs. Training Algorithm

$$\min_{\theta} \underbrace{\frac{1}{T}\sum_{t=1}^{T} \mathbb{E}_{x_0 \sim q(x_0), z_t \sim \mathcal{N}(0,\mathrm{I})} \|z_t - z_\theta(x_t, t)\|_2^2}_{\mathcal{L}_{simple}}$$

$$\hat{\beta}_t = \prod_{i=1}^{t} \alpha_i$$

Training algorithm:

Repeat

$x_0 \sim q(x_0)$ %We sample an image from our data set

$t \sim \mathcal{U}(\{1, \dots, T\})$ %choose randomly a time step t of the forward process

$z_t \sim \mathcal{N}(0, \mathrm{I})$ %sample the noise z_t

$x_t = \sqrt{\hat{\beta}_t} \cdot x_0 + \sqrt{(1 - \hat{\beta}_t)} z_t$ % Get noisy image

$\theta = \theta - lr \cdot \nabla_\theta \mathcal{L}_{simple}$ %Update neural network weights

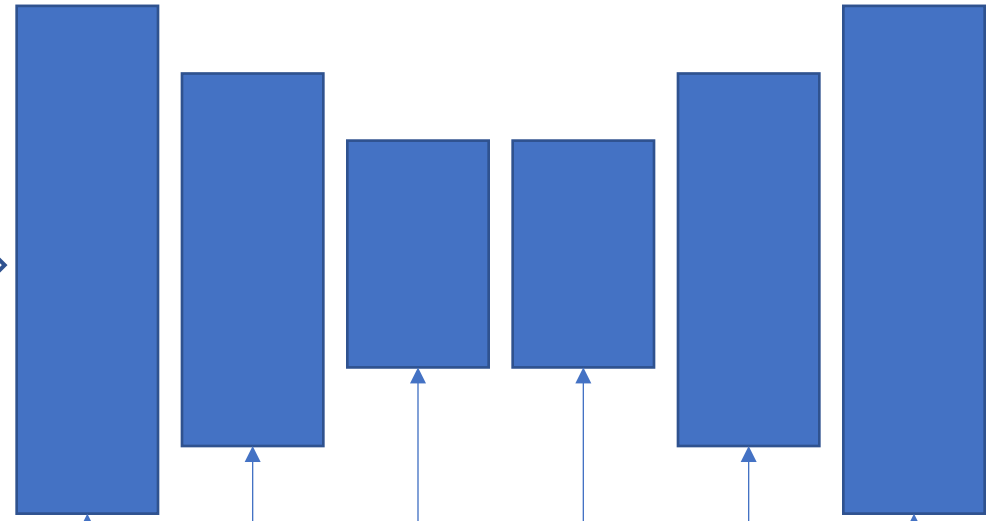
Until convergence

DDPMs. Training objective

UNet-like neural network



x_t



t

$\mu_\theta(x_t, t)$



x_{t-1}

$\sim \mathcal{N}(x_{t-1}, \mu_\theta(x_t, t), \sigma_t^2 \mathrm{I})$

Algorithm 2 Sampling

$\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $t = T, \ldots, 1$ **do**
  $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
  $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
end for
return $\mathbf{x}_0$

Outline

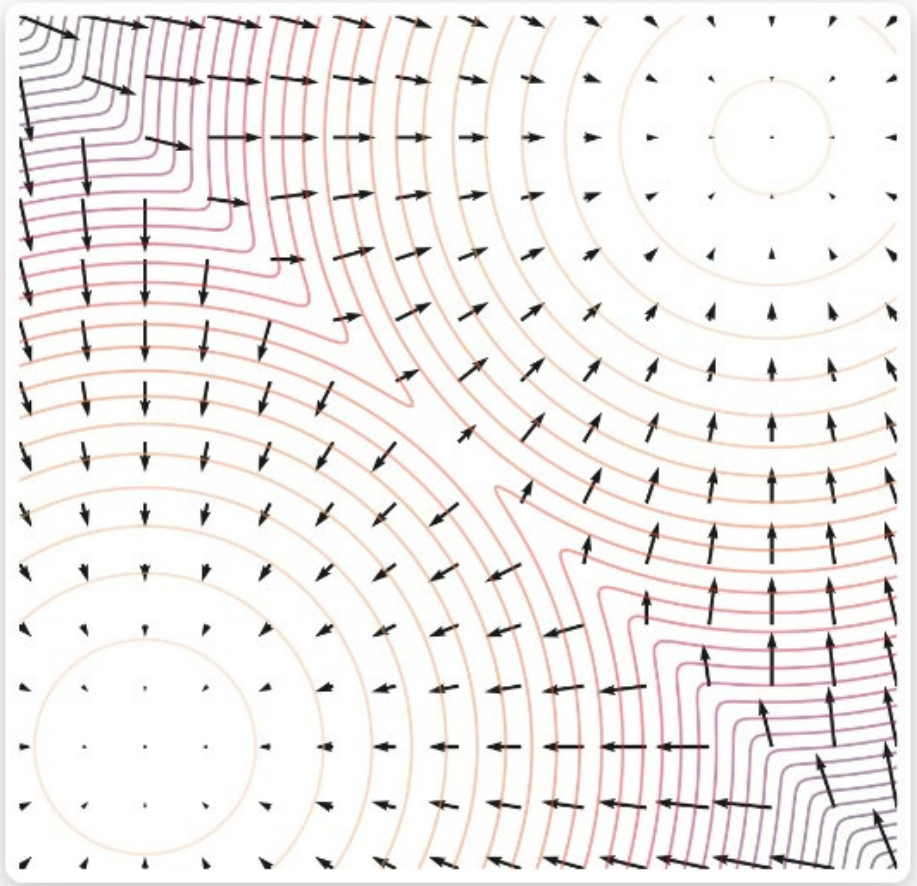
1. Motivation
2. High-level overview
3. Denoising diffusion probabilistic models
4. Noise Conditioned Score Network

Score Function: Second Formulation of Diffusion Models

$$\nabla_x \log p(x)$$

Score-matching refers to the process of modeling the gradient of the log probability density function, also known as the score function.
Langevin dynamics is an iterative process that can draw samples from a distribution using only its score function.

$$x_i = x_{i-1} + \frac{\gamma}{2}\nabla_x \log p(x) + \sqrt{\gamma} \cdot \omega_i$$



- Direction we need to change the input x such that the density becomes greater
- Mixture of two Gaussians in 2D

© Copyright 2022 Yang Song. Powered by Jekyll with al-folio theme.

Naïve score-based model

- **Score**: gradient of the logarithm of the probability density with respect to the input

$$\nabla_x \log p(x)$$

- Langevin dynamics (draw samples from a distribution using only its score function).

$$x_i = x_{i-1} + \frac{\gamma}{2}\nabla_x \log p(x) + \sqrt{\gamma} \cdot \omega_i$$

Step size – controls the magnitude of the update in the direction of the score

Score – estimated by the score network

Noise – random gaussian noise N(0, I)

Naïve score-based model

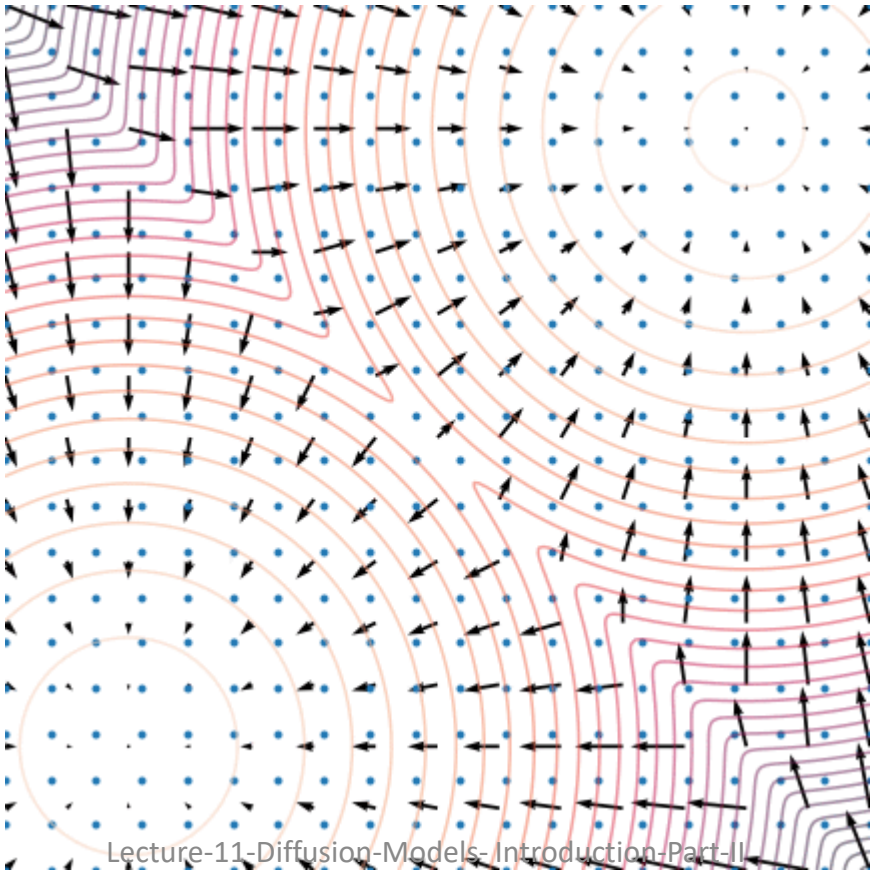
- The score is approximated with a neural network
- Score network is trained using score matching

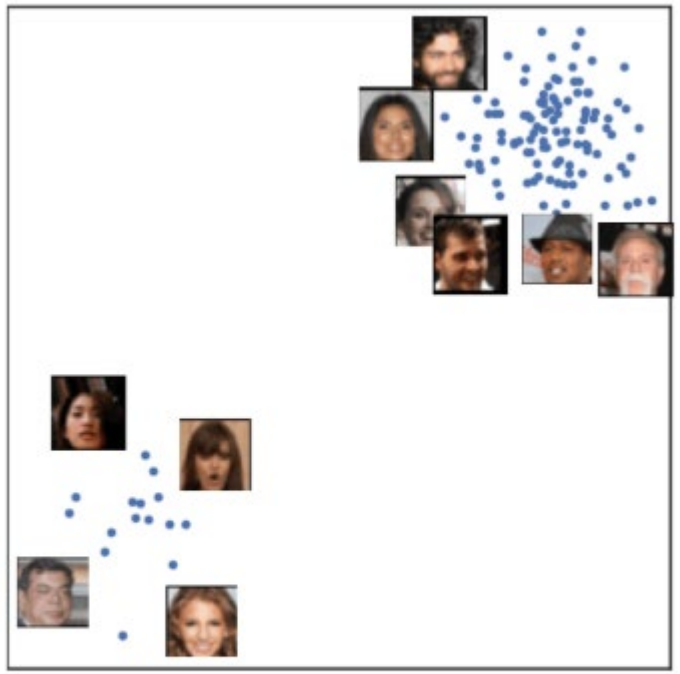
$$\mathbb{E}_{x\sim p(x)}\|s_\theta(x) - \nabla_x \log p(x)\|_2^2$$

  - After training:

$$s_\theta(x) \approx \nabla_x \log p\ (x)$$

Using Langevin dynamics to sample from a mixture of two Gaussians

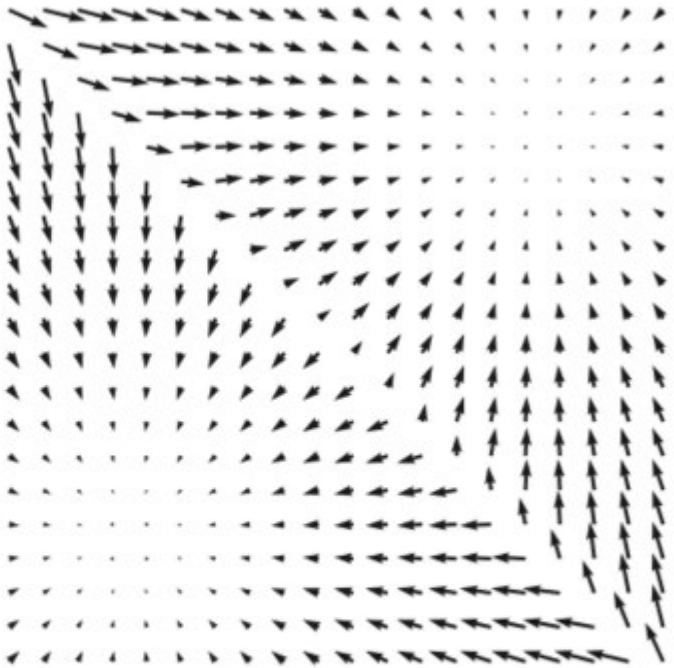




Data samples

$$\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\} \overset{\text{i.i.d.}}{\sim} p(\mathbf{x})$$

score matching



Scores

$$\mathbf{s}_\theta(\mathbf{x}) \approx \nabla_{\mathbf{x}} \log p(\mathbf{x})$$

Langevin dynamics



New samples

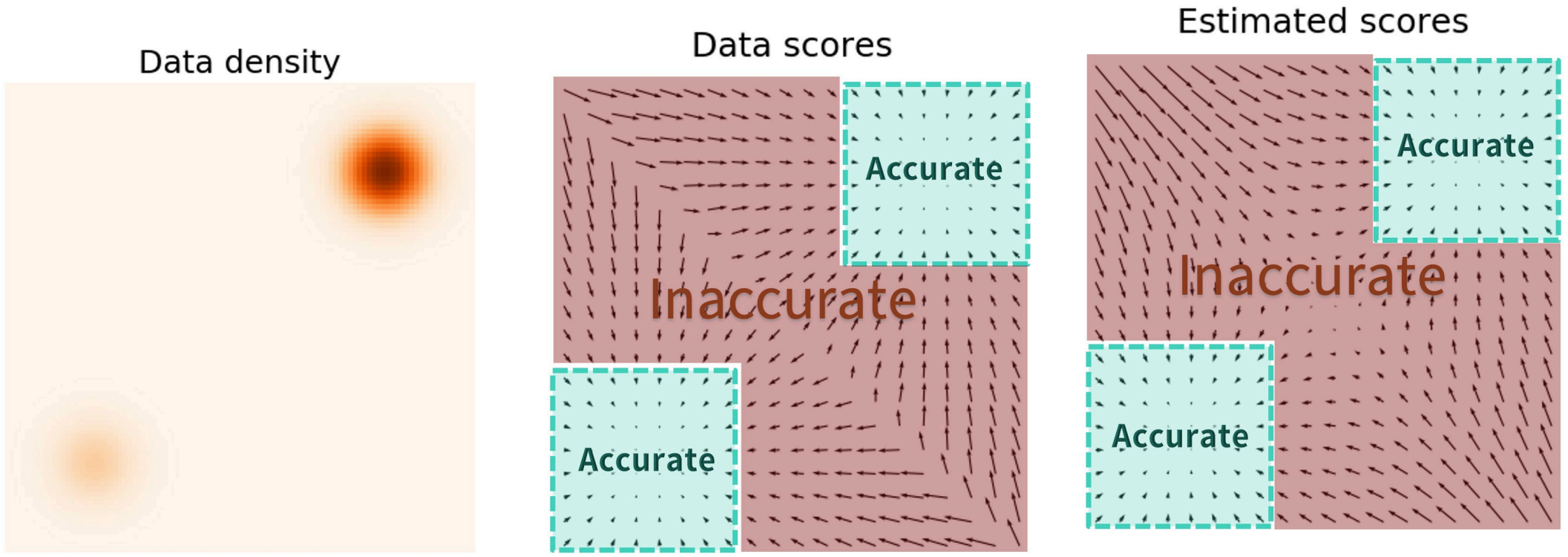
Score-based generative modeling with score matching + Langevin dynamics. Source: Generative Modeling by Estimating Gradients of the Data Distribution

Naïve score-based model. Problems

- Manifold hypothesis: real data resides on low dimensional manifolds
- The score is undefined outside these low dimensional manifolds
- Data being concentrated in regions results in further issues:
  - Incorrectly estimating the score within the low-density regions
  - Langevin dynamics never converging to the high-density region

Credit images Yang Song: https://yangsong.net/blog/2021/score/

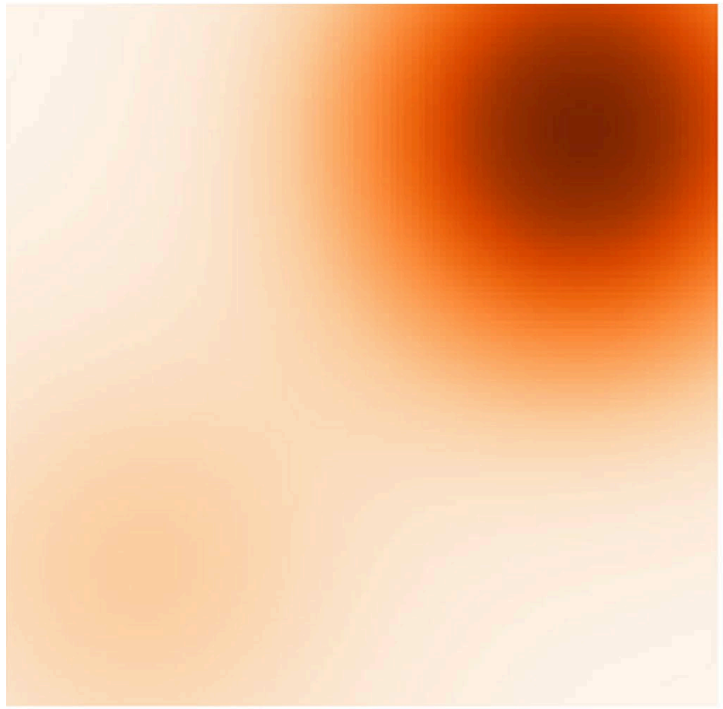
Naïve score-based model. Problems



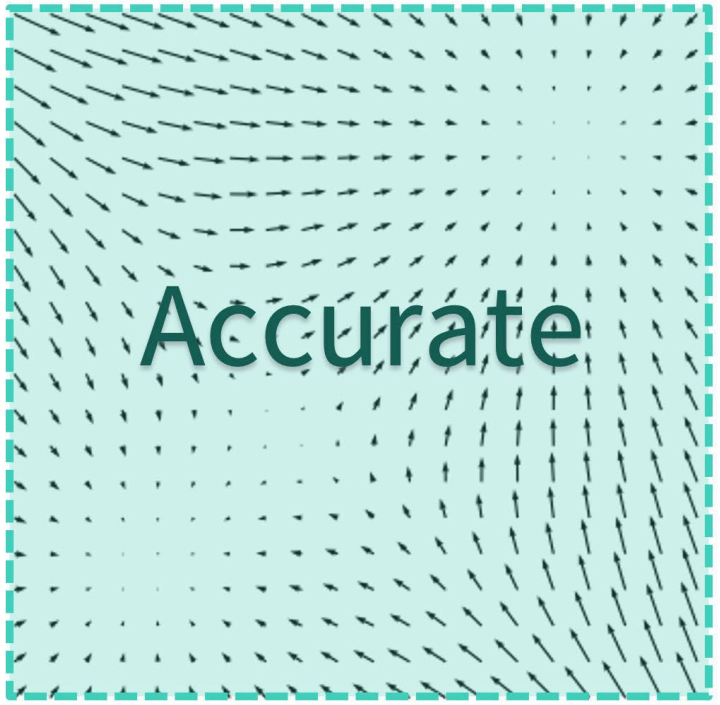
Credit images Yang Song: https://yangsong.net/blog/2021/score/

Noise Perturbation

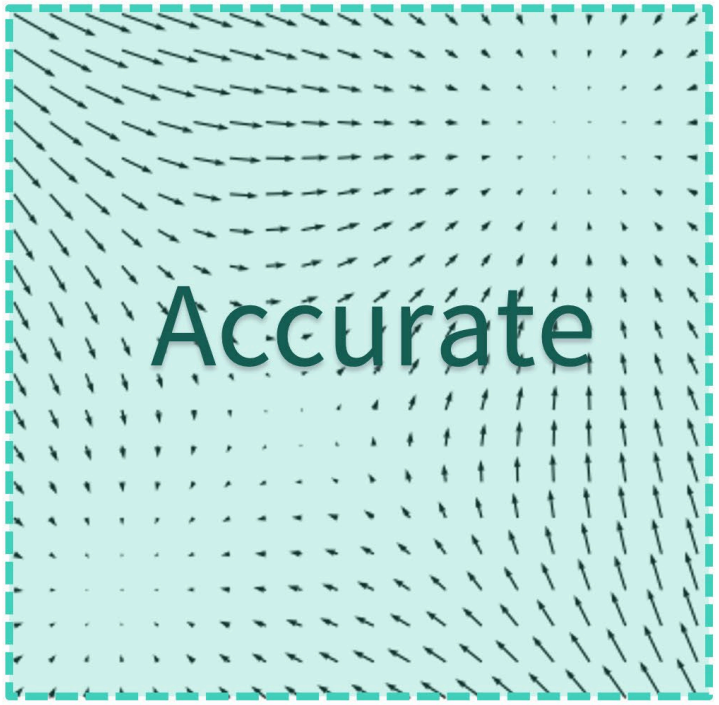
Perturbed density



Perturbed scores

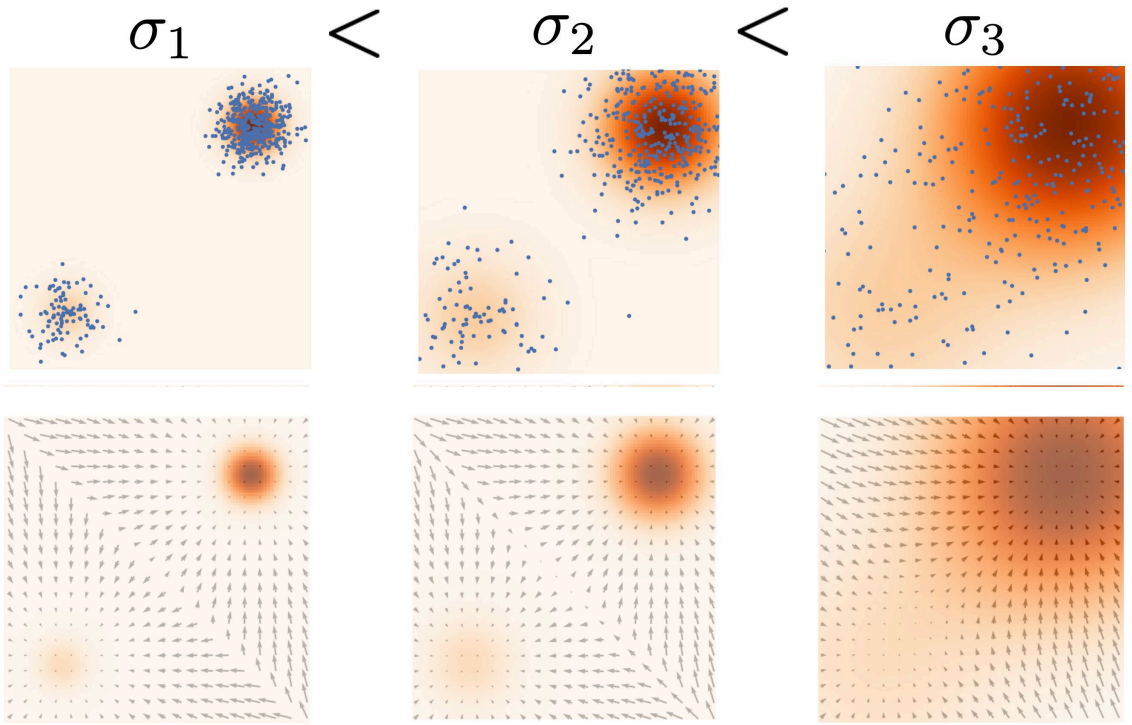


Estimated scores



Noise Conditioned Score Network (NCSNs)

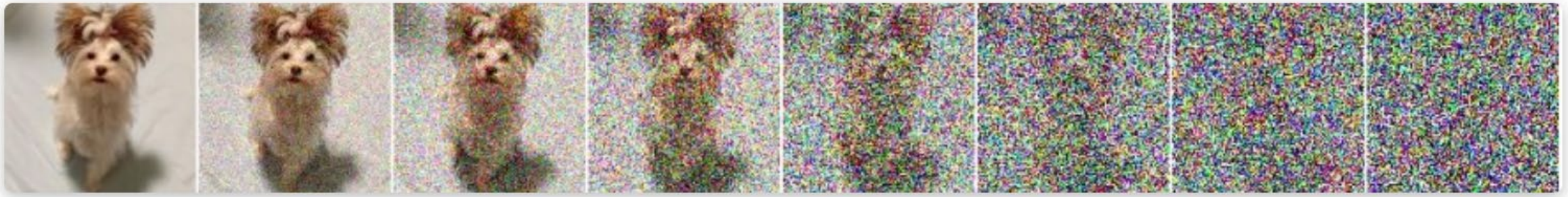
- Solution:
  - ➢Perturb the data with random Gaussian noise at different scales
  - ➢Learn score estimations noisy distributions via a single score network



Credit images Yang Song: https://yang-song.net/blog/2021/score/

Noise Conditioned Score Network (NCSNs)

- Solution:
  - ➢Perturb the data with random Gaussian noise at different scales
  - ➢Learn score estimations noisy distributions via a single score network

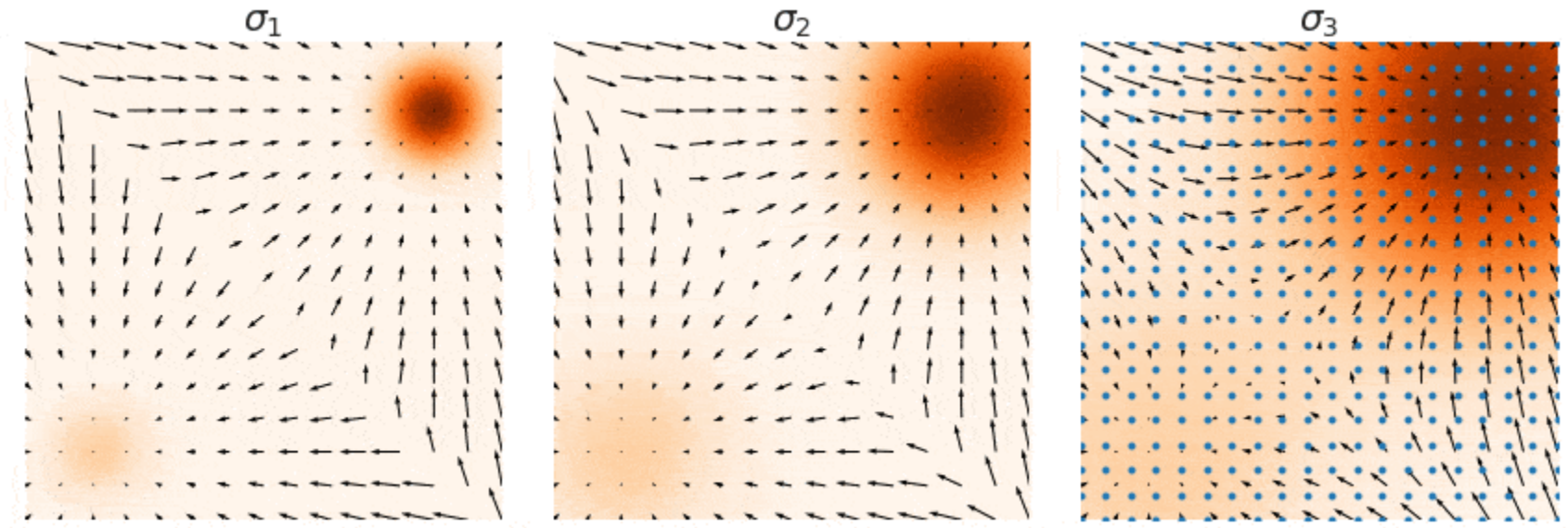


Perturbing an image with multiple scales of Gaussian noise.

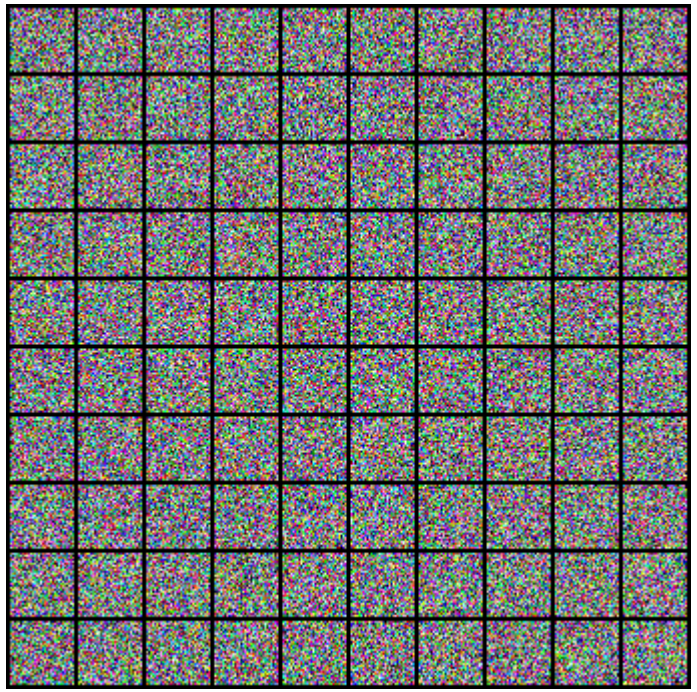
$$\sum_{i=1}^{L} \lambda(i) \mathbb{E}_{p_{\sigma_i}(\mathbf{x})}[\|\nabla_{\mathbf{x}} \log p_{\sigma_i}(\mathbf{x}) - \mathbf{s}_\theta(\mathbf{x}, i)\|_2^2],$$

Credit images Yang Song: https://yang-song.net/blog/2021/score/

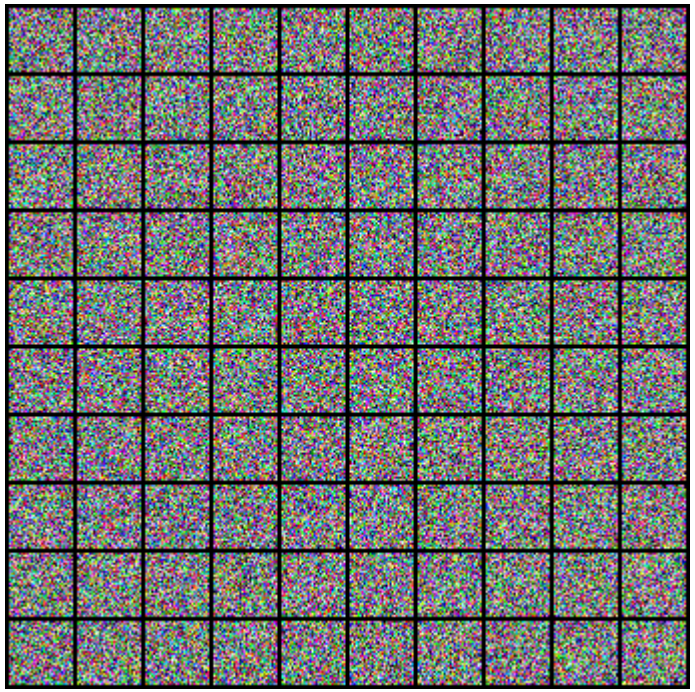
Annealed Langevin dynamics



Annealed Langevin dynamics



CIFAR-10



Celeb-A

Noise Conditioned Score Network (NCSNs)

- Given a sequence of Gaussian noise scales $\sigma_1 < \sigma_2 < \cdots < \sigma_T$ such that:
  - $p_{\sigma_1}(x) \approx p(x_0)$
  - Approximating the true data distribution
  - $p_{\sigma_T}(x) \approx \mathcal{N}(0, I)$
  - Almost equally with the standard gaussian distribution.

- And the forward process i.e. noise perturbation given by:

$$p_{\sigma_t}(x_t|\, x) = \mathcal{N}(x_t; x, \sigma_t^2 \cdot I) = \frac{1}{\sigma_t \cdot \sqrt{2\pi}} \cdot \exp\left(\frac{-1}{2} \cdot \left(\frac{x_t - x}{\sigma_t}\right)^2\right)$$

- The gradient can be written as:

$$\nabla_{x_t} \log p_{\sigma_t}(x_t|\, x) = -\frac{x_t - x}{\sigma_t^2}$$

Noise Conditioned Score Network (NCSNs)

- Training the NCSN with denoising score matching, the following objective is minimized:

$$\mathcal{L}_{dsm} = \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}_{p(x)} \mathbb{E}_{p_{\sigma_t}(x_t|x)} \left\| s_\theta(x_t, \sigma_t) + \frac{x_t - x}{\sigma_t^2} \right\|_2^2$$

Algorithm 1 Training Noise Conditional Score Networks (NCSN) with Denoising Score Matching

Require: Dataset $\mathcal{D}$, noise levels $\{\sigma_1, \sigma_2, \ldots, \sigma_L\}$, number of epochs T, batch size N, learning rate α

Initialize score network model $s_\theta(x, \sigma)$ according to $\mathcal{N}(0, I)$
for epoch $= 1, 2, \ldots, T$ **do**
  for each mini-batch $\{x_i\}_{i=1}^N$ sampled from $\mathcal{D}$ **do**
    Sample σ uniformly from $\{\sigma_1, \sigma_2, \ldots, \sigma_L\}$
    for $i = 1$ to N **do**
      Sample noise $\epsilon_i \sim \mathcal{N}(0, \sigma^2 I)$
      Generate noisy sample: $\tilde{x}_i \leftarrow x_i + \epsilon_i$
      Compute model score: $s_\theta(\tilde{x}_i, \sigma)$
      Compute ground truth score: $-\frac{\tilde{x}_i - x_i}{\sigma^2} = -\frac{\epsilon_i}{\sigma^2}$
    end for
    Compute DSM loss:

$$\mathcal{L}_{DSM} = \frac{1}{2N} \sum_{i=1}^{N} \left\| s_\theta(\tilde{x}_i, \sigma) - \left(-\frac{\epsilon_i}{\sigma^2} \right) \right\|_2^2$$

    Update parameters:

$$\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}_{DSM}$$

  end for
end for
return Trained model $s_\theta(x, \sigma)$.

Algorithm 1 Training Noise Conditional Score Networks (NCSN) with Denoising Score Matching

Require: Dataset $\mathcal{D}$, noise levels $\{\sigma_1, \sigma_2, \ldots, \sigma_L\}$, number of epochs T, batch size N, learning rate α

Initialize score network model $s_\theta(x, \sigma)$ according to $\mathcal{N}(0, I)$
for epoch $= 1, 2, \ldots, T$ **do**
  for each mini-batch $\{x_i\}_{i=1}^N$ sampled from $\mathcal{D}$ **do**
    Sample σ uniformly from $\{\sigma_1, \sigma_2, \ldots, \sigma_L\}$
    for $i = 1$ to N **do**
      Sample noise $\epsilon_i \sim \mathcal{N}(0, \sigma^2 I)$
      Generate noisy sample: $\tilde{x}_i \leftarrow x_i + \epsilon_i$
      Compute model score: $s_\theta(\tilde{x}_i, \sigma)$
      Compute ground truth score: $-\frac{\tilde{x}_i - x_i}{\sigma^2} = -\frac{\epsilon_i}{\sigma^2}$
    end for
    Compute DSM loss:

$$\mathcal{L}_{DSM} = \frac{1}{2N} \sum_{i=1}^{N} \left\| s_\theta(\tilde{x}_i, \sigma) - \left(-\frac{\epsilon_i}{\sigma^2} \right) \right\|_2^2$$

    Update parameters:

$$\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}_{DSM}$$

  end for
end for
return Trained model $s_\theta(x, \sigma)$.

Noise Conditioned Score Network (NCSNs). Sampling

Annealed Langevin dynamics

Parameters:

N – number of iterations for Langevin dynamics

$\sigma_0 < \sigma_1 < \ldots < \sigma_T$ - noise scales

$\{\gamma_t | \ t = 1, T\}$ - update magnitude

Algorithm:

$x_T^0 \sim \mathcal{N}(0, I)$

for t $= T, 1$ do:

  for i $= 1, N$ do:

    $z_t \sim \mathcal{N}(0, I)$

    $x_t^i = x_t^{i-1} + \frac{\gamma_t}{2} \cdot s_\theta\left(x_t^{i-1}, \sigma_t\right) + \sqrt{\gamma_t} z_t$

  $x_{t-1}^0 = x_t^N$

return x_0

Noise Conditioned Score Network (NCSNs). Sampling

Annealed Langevin dynamics

Parameters:

N – number of iterations for Langevin dynamics

$\sigma_0 < \sigma_1 < \ldots < \sigma_T$ - noise scales

$\{\gamma_t | \ t = 1, T\}$ - update magnitude

Algorithm:

$x_T^0 \sim \mathcal{N}(0, I)$; %sample some standard gaussian noise

for t $= T, 1$ do: %start from the largest noise scale, which is denoted by the time step

  for i $= 1, N$ do: %for N iterations execute the Langevin dynamics updates

    $z_t \sim \mathcal{N}(0, I)$ % get noise

    $x_t^i = x_t^{i-1} + \frac{\gamma_t}{2} \cdot s_\theta\left(x_t^{i-1}, \sigma_t\right) + \sqrt{\gamma_t} z_t$ % update

  $x_{t-1}^0 = x_t^N$ % next iteration

return x_0

Similarities of DDPM vs NCSN. Losses

DDPM: $\mathcal{L}_{simple} = \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}_{x_0 \sim p(x_0), z_t \sim \mathcal{N}(0,\mathrm{I})} \|z_\theta(x_t, t) - z_t\|_2^2$

NCSN: $\mathcal{L}_{dsm} = \frac{1}{T}\sum_{t=1}^{T} \lambda(\sigma_t) \mathbb{E}_{x_0 \sim p(x_0),\, x_t \sim p_{\sigma_t}(x_t|x_0)} \left\| s_\theta(x_t, \sigma_t) + \frac{x_t - x_0}{\sigma_t^2} \right\|$

We can rewrite the noise z_t, as follows:

$$x_t \sim \mathcal{N}(x_t;\, x_0, \sigma_t^2 \mathrm{I}) \Rightarrow x_t = x_0 + \sigma_t \cdot z_t,\, z_t \sim \mathcal{N}(0, I) \Rightarrow z_t = \frac{x_t - x_0}{\sigma_t}$$

- So, $s_\theta(x_t, \sigma_t)$ learns to approximate a scaled negative noise $\frac{-z_t}{\sigma_t}$.

DDPM vs NCSN. Sampling

DDPM: $x_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\widehat{\beta}_t}} z_\theta(x_t, t)\right) + \sqrt{\beta_t} \cdot z_t$

- Iterative updates are based on subtracting some form of noise from the noisy image.

NCSN: $x_t^i = x_t^{i-1} + \frac{\gamma_t}{2} \cdot s_\theta\left(x_t^{i-1}, \sigma_t\right) + \sqrt{\gamma_t} \cdot z_t$

- This is true also for NCSN because $s_\theta\left(x_t^{i-1}, \sigma_t\right)$ approximates the negative of the noise.

$$\mathcal{L}_{dsm} = \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}_{p(x)} \mathbb{E}_{p_{\sigma_t}(x_t|x)} \left\| s_\theta(x_t, \sigma_t) + \frac{x_t - x}{\sigma_t^2} \right\|_2^2$$

$$\mathcal{L}_{dsm} = \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}_{p(x)} \mathbb{E}_{p_{\sigma_t}(x_t|x)} \left\| s_\theta(x_t, \sigma_t) - (-\frac{x_t - x}{\sigma_t^2}) \right\|_2^2$$

$$\underbrace{\min_\theta \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}_{x_0 \sim p(x_0), z_t \sim \mathcal{N}(0,\mathrm{I})} \|z_t - z_\theta(x_t, t)\|_2^2}_{\mathcal{L}_{simple}}$$

$$z_t = \frac{x_t - x_0}{\sigma_t}$$

- Therefore, the generative processes defined by NCSN and DDPM are very similar.

Outline

1. Motivation
2. High-level overview
3. Denoising diffusion probabilistic models
4. Noise Conditioned Score Network
5. Stochastic Differential Equations
6. Conditional Generation
7. Research directions

Conditional Generation

- So far, we have discussed unconditioned generation

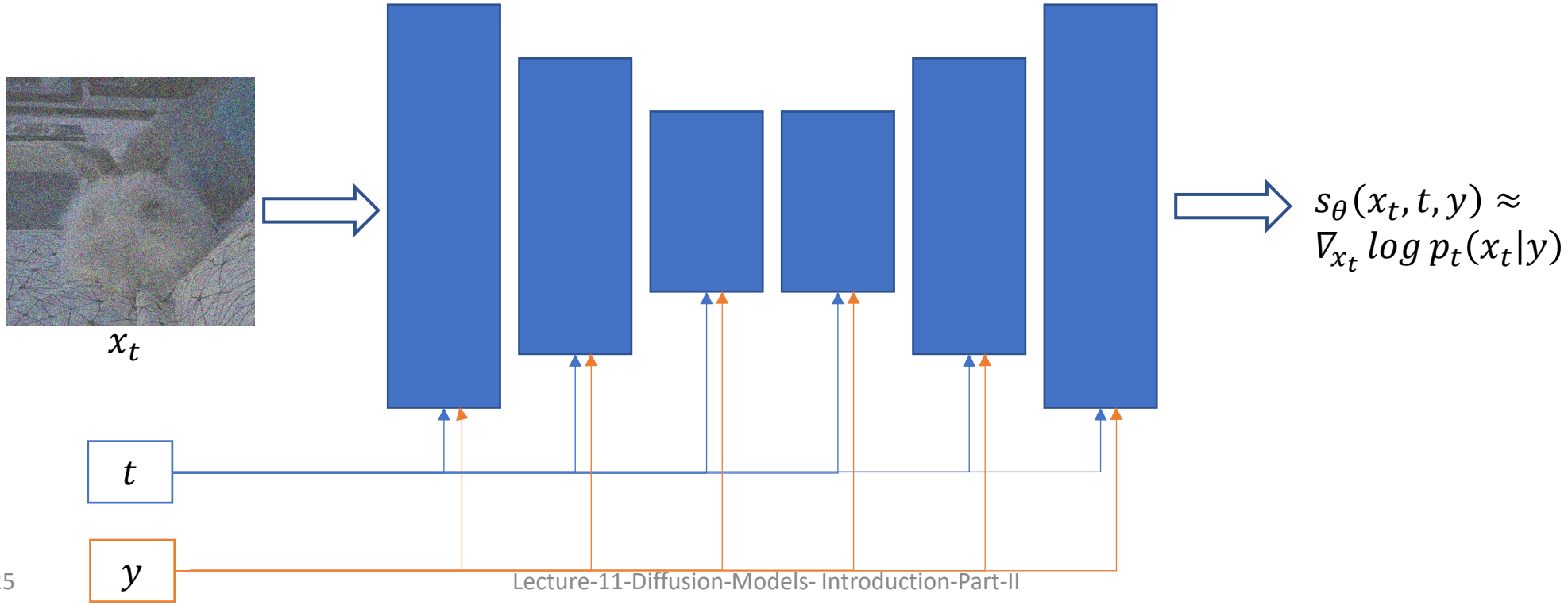
- Conditional generation
  - Class Condition
  - Text Condition
  - Image Condition

Conditional generation.

Diffusion models estimate the score function, $\boldsymbol{\nabla_{x_t} log\, p_t(x_t)}$ to sample from a distribution $\boldsymbol{p(x)}$.

Sampling from $\mathbf{p}(\boldsymbol{x}|\boldsymbol{y})$ requires the score function of this probability density, $\boldsymbol{\nabla_{x_t} log\, p_t(x_t|y)}$; $\boldsymbol{y}$ is condition.

Solution 1. **Conditional training:** train the model with an additional input y to estimate $\boldsymbol{\nabla_{x_t} log\, p_t(x_t|y)}$.



Conditional generation. Classifier Guidance

Diffusion models estimate the score function, $\boldsymbol{\nabla_{x_t} log\, p_t(x_t)}$ to sample from a distribution $\boldsymbol{p(x)}$.

Sampling from $\mathbf{p}(\boldsymbol{x}|\boldsymbol{y})$ requires the score function of this probability density, $\boldsymbol{\nabla_{x_t} log\, p_t(x_t|y)}$.

Solution 2. **Classifier guidance:**

Bayes rule:

$$p_t(x_t|y) = \frac{p_t(y|x_t) \cdot p_t(x_t)}{p_t(y)} \Leftrightarrow$$

Logarithm:

$$\log p_t(x_t|y) = \log p_t(y|x_t) + \log p_t(x_t) - \log p_t(y) \Leftrightarrow$$

Gradient:

$$\nabla_{x_t} \log p_t(x_t|y) = \nabla_{x_t} \log p_t(y|x_t) + \nabla_{x_t} \log p_t(x_t) - \nabla_{x_t} \log p_t(y) \Leftrightarrow$$

$$\boldsymbol{\nabla_{x_t} log\, p_t(x_t|y) = \nabla_{x_t} log\, p_t(y|x_t) + \nabla_{x_t} log\, p_t(x_t)}$$

Classifier

Unconditional diffusion model

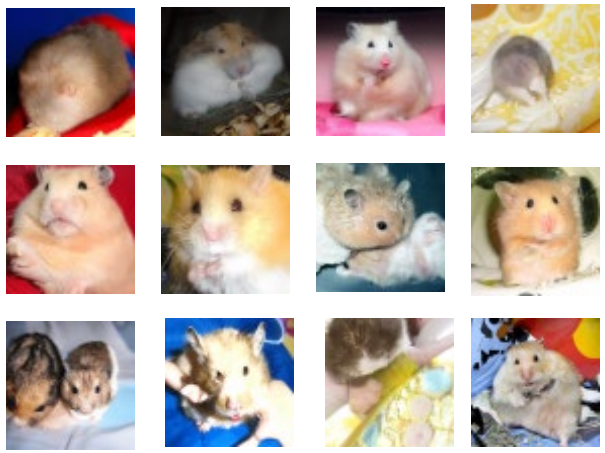
Conditional generation. Classifier Guidance

Solution 2. **Classifier guidance:**

Guidance weight

$$\nabla_{x_t} log\, p_t(x_t|y) = s \cdot \nabla_{x_t} log\, p_t(y|x_t) + \nabla_{x_t} log\, p_t(x_t)$$

Hamster



$s = 1$



$s = 10$

Problem

- Need to have good gradients estimates at each step of denoising process
    - Need a classifier that is robust to noise added in the image.
    - Training of the classifier on noisy data, which can be problematic..

Conditional generation. **Classifier-free Guidance**

Solution 3. **Classifier-free guidance**

$$\nabla_{x_t} log\, p_t(x_t|y) = s\ \cdot \nabla_{x_t} log\, p_t(y|x_t) + \nabla_{x_t} log\, p_t(x_t)$$

Bayes rule:

$$p_t(y|x_t) = \frac{p_t(x_t|y)\ \cdot p_t(y)}{p_t(x_t)}$$

Logarithm:

$$\log p_t(y|x_t) = \log p_t(x_t|y) - \log p_t(x_t) + \log p_t(y)$$

Gradient

$$\nabla_{x_t} log\, p_t(y|x_t) = \nabla_{x_t} log\, p_t(x_t|y) - \nabla_{x_t} \log p_t(x_t)$$

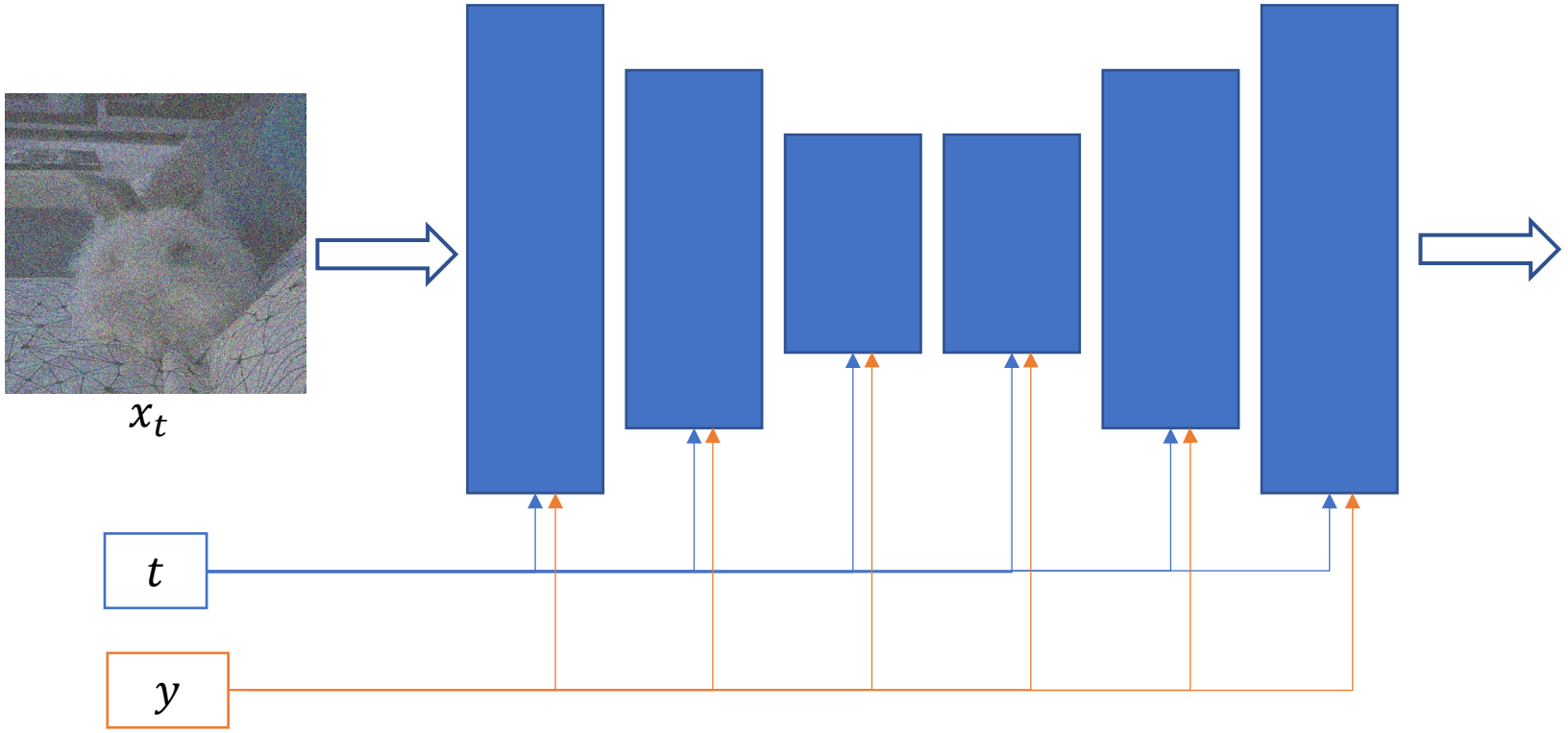
from above $\nabla_{x_t} log\, p_t(x_t|y) = s\ \cdot \nabla_{x_t} log\, p_t(y|x_t) + \nabla_{x_t} log\, p_t(x_t)$

$$\nabla_{x_t} log\, p_t(x_t|y) = s\ \cdot (\nabla_{x_t} log\, p_t(x_t|y) - \nabla_{x_t} \log p_t(x_t)) + \nabla_{x_t} log\, p_t(x_t)$$

$$\boldsymbol{\nabla_{x_t} log\, p_t(x_t|y) = s\ \cdot \nabla_{x_t} log\, p_t(x_t|y) + (1-s) \cdot \nabla_{x_t} log\, p_t(x_t)}$$

Learned by a single model

Conditional generation. Classifier-free Guidance



Conditional generation. Classifier-free Guidance

